Confident Collaborative Metric Learning

Ryo Matsui
Tokyo Institute of Technology
Tokyo, Japan
Email:matsui.r.ae@m.titech.ac.jp

Taketo Naito
SMN, Corp.
Tokyo, Japan
Email:taketo_naito@so-netmedia.jp

Suguru Yaginuma
MC Digital, inc.
Tokyo, Japan
Email:suguru@mcdigital.jp

Kazuhide Nakata
Tokyo Institute of Technology
Tokyo, Japan
Email:nakata.k.ac@m.titech.ac.jp

2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00038

***Abstract*—Modern recommendation systems use embedding for secondary applications and implicit feedback data for learning. In recent years, collaborative metric learning (CML), a method that can precisely capture the relationship between users and items, has been developed for the first requirement. However, CML with implicit feedback data suffers from noisy-label issues. This is mainly because CML and the related works only consider few types of noise. To overcome these limitations, we develop a method for estimating the noise rate and excluding extremely noisy user–item pairs from the data prior to learning CML. Experimental results show that the proposed method significantly outperforms existing methods in terms of ranking metrics.**

I. INTRODUCTION

Recommendation systems can predicted the relevance of user–item pairs based on the embeddings of users and items. Furthermore, modern recommendation systems are required to possess two significant features.

The first requirement is the use of embeddings for secondary applications such as user–user or item–item recommendations, in addition to the typical user–item recommendations, including friend recommendations and similar item recommendations [1], [2], [3], [4]. Thus, it is desirable to capture the user–user and item–item relationships for modern web services. The standard method for recommendations with embeddings is the matrix factorization (MF) based model; however, previous research has indicated that this approach may fail to capture finer-grained preference information owing to the loss function of MF [5], [6], [7]. This method has been optimized for predicting user–item interactions, but it neglects the similarity between user–user and item–item pairs. To solve this issue, Hsieh et al. developed an algorithm called collaborative metric learning (CML) [5]. CML learns the distance metric of the joint vector space of users and items. As described in Section 2, the triangle inequality of distance metrics enables CML to learn the relationships between user–user pairs and item–item pairs. Given that this approach addresses the user–user / item–item relationship problem, it has been applied and further extended [8], [9], [10], [11], [12], [13].

The second requirement is the use of implicit feedback. In today's web services, it is easy to collect log data without any active action by the user [14]. Thus, implicit feedback, such as clicking on a web page, has been widely used in recommendation systems [15], [16], [17], [18], [19]. However, this approach of using implicit feedback suffers from noisy-label issues. Prior works [20], [21], [22] have indicated differences between implicit feedback and actual user satisfaction. Furthermore, there are two types of noise. The first is **1) positive noise (false-positive)**: An event in which a user interacts with an item but is not interested in the item. Such an event occurs because the first impression of a caption or headline causes users to click on it [23], [22]. This type of noise can lead to low-quality recommendations because the system tends to suggest an item that the user does not like. The second type is **2) negative noise (false-negative)**: An event in which users miss their favorite items. This is also referred to as the positive-unlabeled problem [24], [25], [19]. In other words, a lack of interaction does not always indicate a negative response from users. Thus, the system should not learn such potentially positive but unobserved items as entirely negative items.

In summary, to build a modern recommendation system, we must solve the user–user / item–item relationship problem and address these noisy-label issues associated with implicit feedback.

Tran et al. [26] addressed both of the abovementioned problems by considering noise during the training of CML. As CML is based on triplet loss, it repeats training with a batch of triplets selected via uniformly sampling. This, however, is a disadvantage of CML. Uniform sampling is inefficient for learning user–item relationships because it selects noisy data with a certain probability. To resolve this problem, [26] proposed a two-stage negative sampling strategy weighted by the noise rate of negative items.

However, there are two main problems with [26].

The first problem is that weighted sampling might not be optimal for CML. Although weighted sampling is less likely to select noisy negative pairs, a negative sample set that includes even one noisy pair will degrade the overall performance owing to the triangle inequality. Therefore, extremely noisy

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00038

negative pairs should not be used for training.

The second problem is that previous works do not consider positive noise. Although the extension of negative sampling can reduce negative noise, positive noise also has a significant harmful effect on learning. Therefore, extremely noisy positive pairs should also not be used for training, in contrast to negative pairs.

To solve these problems, we propose a method that eliminates user–item pairs with extreme noise prior to CML. To judge extremely noisy pairs, we estimate the noise rate for positive and negative noise in advance by using user and item data. This makes it easy to handle positive and negative noise. This process also prevents CML in the wrong direction owing to the triangle inequality. Confident learning [27], which is a state-of-the-art image classification method with noisy labels, supports this method of noise rate estimation using noisy data and by cleaning the data. In our experiments, this method is applied to the datasets whose training data include noise but the test data include almost no noise. The results indicate that the proposed method improves three ranking metrics by more than 10%.

Beyond the improvement in the metrics for usual user–item recommendations, we investigated the method for capturing user–user and item–item relationships. After calculating these metrics, we visualized how close the embeddings of users and items with similar features are and evaluate how accurately the SVM classifies the features with them. The result shows that, although the naive methods did not always represent the features well, the proposed method alleviated this problem. In addition, the error rate for classification improved from 7.3% to 5.6% with respect to user gender and from 1.7% to 0.7% with respect to the target gender for clothing.

II. PRELIMINARIES

In this section, we describe the formulation of a recommendation system with implicit feedback and summarize the related works.

A. Problem Formulation & Notation

Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ be the set of items. We assume that the system recommends the top K items, $i_1, i_2, \ldots, i_k \in \mathcal{I}$, that are relevant to user $u \in \mathcal{U}$ when the user arrives. We consider $Y_{u,i} = 1$ if an interaction between user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ occurs before the arrival of this user, and $Y_{u,i} = -1$ otherwise. The set of all interactions collected at this point is denoted by $\mathcal{S} = \{(u,i)|Y_{u.i} = 1\}$, and the set of all user–item pairs not included in this set is denoted by $\mathcal{D} = \{(u,i)|Y_{u.i} = -1\}$. The system uses this information to predict the relevance of users and items. In collaborative filtering, the system learns embeddings $\mathbf{v}_u$ and $\mathbf{v}_i$ for users and items, respectively, and recommends items based on the values of similarity $\mathrm{Sim}(\mathbf{v}_u, \mathbf{v}_i)$. Note that the observed interaction, $Y_{u,i}$, is not necessarily equal to the relevance, $R_{u,i} \in \{1, -1\}$, between a true user and an item. We assume that relevance $R_{u,i}$ is not observable and that $Y_{u,i}$ is observed with noise. Therefore, it is necessary to consider noise when learning the embeddings.

B. Matrix Factorization (MF)

Collaborative filtering based on matrix factorization is commonly used in recommendation systems. This method considers a similarity function, $\mathrm{Sim}(u,i) = \mathbf{v}_u^T \mathbf{v}_i$. Hu et al. formulated the weighted matrix factorization on the basis of $c_{u,i}$, which is the reliability of $Y_{u,i}$, as follows [20]:

$$\min_{\mathbf{v}_*} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} c_{u,i} (Y_{u,i} - \hat{Y}_{u,i})^2, \text{where } \hat{Y}_{u,i} = \mathbf{v}_u^T \mathbf{v}_i$$

C. Collaborative Metric Learning (CML)

The above matrix-factorization-based methods mainly try to represent the relationship between the inner product of user and item embeddings. However, the inner products of user–user pairs and item–item pairs do not represent the strength of their relationships. Hsieh et al.[5] pointed out that this property strongly affects the quality and interpretability of recommendations and proposed CML as a method of solving this problem.

CML learns embeddings by setting the loss function as

$$\mathcal{L}^{\mathrm{triplet}}(\mathbf{v}_*) = \sum_{(u,i) \in \mathcal{S}} \sum_{j \in \mathcal{D}_u} w_{u,i} \left[m + d(\mathbf{v}_u, \mathbf{v}_i)^2 - d(\mathbf{v}_u, \mathbf{v}_j)^2 \right]_+ \quad (1)$$

and solving for

$$\min_{\mathbf{v}_*} \mathcal{L}^{\mathrm{triplet}}(\mathbf{v}_*), \quad \text{s.t. } \|\mathbf{v}\|^2 \leq 1, \forall \mathbf{v}_*.$$

Here, d represents the Euclidean distance, which is the similarity function in CML, $\mathcal{D}_u = \{i|Y_{u,i} = -1\}$ is a negative item set for a user u, $w_{u,i}$ is the weight, and m represents the maximum margin. The triangle inequality of the distance function enables CML to capture user–user / item–item relationships. This property comes from

$$\text{user–user pair} : d(u_1, i_1) + d(u_2, i_1) > d(u_1, u_2),$$
$$\text{item–item pair} : d(u_1, i_1) + d(u_1, i_2) > d(i_1, i_2).$$

If we observe $(u_1, i_1), (u_1, i_2), (u_2, i_1)$ as positive and reduce the distance between these pairs, the distance of user–user pair (u_1, u_2) and item–item pair (i_1, i_2) are also reduced. Moreover, the triangle inequality allows CML to learn unobserved user–item pairs as relevant pairs from

$$d(u_1, i_1) + d(u_1, i_2) + d(u_2, i_1) > d(i_1, i_2) + d(u_2, i_1) > d(u_2, i_2).$$

As shown in Figure 1, the distance of unobserved pair (u_2, i_2) are also reduced when $Y_{u_1,i_1} = Y_{u_1,i_2} = Y_{u_2,i_1} = 1$.

However, it is practically difficult to find the global optima of $\mathcal{L}^{\mathrm{triplet}}$ for a large number of combinations of $\mathcal{S}$ and $\mathcal{D}$. Then, we apply minibatch stochastic gradient descent for the loss function defined below with positive sample $\mathcal{B} \subset \mathcal{S}$ and negative sample $\mathcal{N}_{u,i} \subset \mathcal{I}$ to find the local optima.

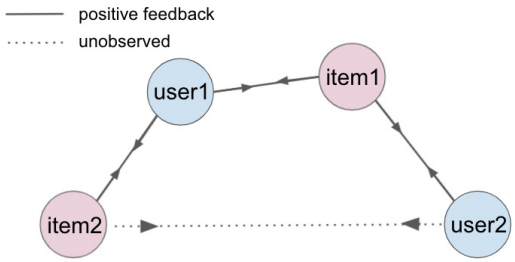

Fig. 1. Illustration of the effect of triangle inequality in CML.

$$\mathcal{L}^{\text{batch}}(\mathbf{v}_*, \mathcal{B}, \{\mathcal{N}_{u,i}\}_{(u,i)\in\mathcal{B}}) = \sum_{(u,i)\in\mathcal{B}} w_{u,i}[d^2(\mathbf{v}_u, \mathbf{v}_i) + \min_{j\in\mathcal{N}_{u,i}} d^2(\mathbf{v}_u, \mathbf{v}_j) + m]_+ \quad (2)$$

That is, we repeatedly sample minibatches and update parameters in the direction of decreasing this loss function. Here we call the sampling of $\mathcal{B}$ positive sampling and the sampling of $\mathcal{N}_{u,i}$ negative sampling.

D. Weighted Negative Sampling

The abovementioned method neglects the causes of noise. In contrast, weighted negative sampling additionally considers the **negative noise** caused by the popularity of items. This method weights the negative sampling probability using the number of observed interactions of an item in a dataset. Previous studies on representation learning have shown that this method is effective [28], [29]. If the number of observations in a dataset is $f(j)$, the sampling probability of item j in negative sampling, $\Pr(j)$, is expressed as follows:

$$\Pr(j) = \frac{f(j)^\beta}{\sum_{j'} f\,(j')^\beta}$$

β represents the hardness of the weights. We can interpret this method as the assumption that $P(R_{u,i} = -1|Y_{u,i} = -1)$ depends on the popularity of an item.

E. 2-stage Negative Sampling

Tran et al. proposed a negative sampling strategy that adds one more step to weighted negative sampling [26]. Specifically, the two steps are as follows:

1) Select a negative sample candidate, $\mathcal{C}_{u,i} \subset \mathcal{I}$, in weighted negative sampling.
2) Select a highly informative sample, $\mathcal{N}_{u,i} \subset \mathcal{C}_{u,i}$, among the candidates on the basis of the probability defined below.

$$\Pr(j \mid \mathbf{v}_i) \propto \begin{cases} \frac{1}{p\left(\mathbf{v}_i^T\mathbf{v}_j = s\right)}, & 0 \le s \le 1 \\ 0, & \text{otherwise} \end{cases}$$

$$p\left(\mathbf{p}_1^T\mathbf{p}_2 = s\right) = \begin{cases} \frac{\left(1-s^2\right)^{\frac{d-1}{2}-1}}{\text{Beta}\left(\frac{d-1}{2}, \frac{1}{2}\right)} & \text{if } -1 \le s \le 1 \\ 0 & \text{otherwise} \end{cases}$$

This method also considers only the false-negative noise caused by the popularity of an item.

III. PROPOSED METHOD

Weighted negative sampling and 2-stage sampling have two problems in terms of noise. The first is that weighted sampling does not completely eliminate the harmful effects of **negative noise**. The second is that they cannot handle the **positive noise**. To address these problems, we propose a method of cleaning data based on confident learning [27].

Confident learning is a method of cleaning data using the results of learning on noisy data. This method first builds a model that predicts the probability of belonging to each label, $\hat{p}(y|\mathbf{x}_i) = f(\mathbf{x}; \Theta), f : \mathbb{R}^d \to [0,1]^M$. It uses dataset $\mathcal{D} = (\mathbf{x}_i, \tilde{y}_i) \in \mathbb{R}^d \times \{1, 2, \ldots, M\}$, whose instances are features and noisy labels. Next, it removes samples with a high noise rate $\rho_{j,\mathbf{x}_i} = \hat{p}(y = j|\mathbf{x}_i) - \hat{p}(y = \tilde{y}_i|\mathbf{x}_i)\}, (j \neq \tilde{y}_i)$ from $\mathcal{D}$. Finally, it performs training again using clean data. We consider the same steps for CML with implicit feedback.

A. Estimate Noise Rate

First, we estimate the noise rate. The positive noise rate, $\rho^+_{u,i}$, in the positive sample, $\mathcal{S} = \{(u,i)|Y_{u,i} = 1\}$, and the negative noise rate, $\rho^-_{u,i}$, in the negative sample, $\mathcal{D} = \{(u,i)|Y_{u,i} = -1\}$, can be expressed as

$$\rho^+_{u,i} = \hat{P}(Y_{u,i} = -1) - \hat{P}(Y_{u,i} = 1) = 1 - 2\hat{P}(Y_{u,i} = 1),$$
$$\rho^-_{u,i} = \hat{P}(Y_{u,i} = 1) - \hat{P}(Y_{u,i} = -1) = 2\hat{P}(Y_{u,i} = 1) - 1,$$

similar to confident learning. Thus, we can calculate these noise rates by estimating $\hat{P}(Y_{u,i} = 1)$ for all user–item pairs, $(u,i) \in \mathcal{U} \times \mathcal{I}$.

To estimate noise rates, we estimate $\hat{P}(Y_{u,i} = 1)$ using the given label, $Y_{u,i}$. In other words, for any model, $f : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$, parametrized by Θ, certain algorithms learn Θ as $\hat{P}(Y_{u,i} = 1) = \sigma(f(u,i;\Theta))$. Note that σ is a sigmoid function. Assuming that $P(Y_{u,i} = 1)$ follows a binomial distribution, this problem is the minimization problem of

$$L(\Theta) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} - Y_{u,i}\log\sigma(f(u,i;\Theta)) - (1 - Y_{u,i})\log\sigma(-f(u,i;\Theta)). \quad (3)$$

This model and algorithm are the same as those used in standard recommendation systems, such as logistic matrix factorization (LMF) [30] and factorization machines [31]. For example, LMF defines a model, $f(u,i;\mathbf{v}_*,b_*) = \mathbf{v}_u^T\mathbf{v}_i + b_u + b_i$, by embedding $\mathbf{v}_u \in \mathbb{R}^d\ (\forall u \in \mathcal{U}), \mathbf{v}_i \in \mathbb{R}^d\ (\forall i \in \mathcal{I})$, and a bias term, $b_u \in \mathbb{R}\ (\forall u \in \mathcal{U}), b_i \in \mathbb{R}\ (\forall i \in \mathcal{I})$. It uses a gradient descent method, such as Adam[32], to find the local optima. This procedure allows us to compute $\hat{P}(Y_{u,i} = 1)$ and the noise rate, $\rho^+_{u,i}, \rho^-_{u,i}$.

B. Cleaning Data & Learning CML Model

We use $\rho^+_{u,i}, \rho^-_{u,i}$ calculated in the previous section to determine the user–item pairs to be removed during positive and negative sampling in CML. Here, we explain how to remove noisy user–item pairs from $\mathcal{S}$.

First, we set the proportion, p, of data to be deleted and calculate the threshold, t_+, as

$$t_+ = \mathrm{Percentile}(\{\rho^+_{u,i}\}_{(u,i)\in\mathcal{S}}, p).$$

Here, $\mathrm{Percentile}(\mathcal{A}, p)$ denotes the top p percentile of scalar-element set $\mathcal{A}$. Then, the set of clean user–item pairs, $\mathcal{S}'$, is defined as

$$\mathcal{S}' = \{(u,i)|(u,i) \in \mathcal{S} \wedge \rho^+_{u,i} \leq t_+\}.$$

Removing a sample from $\mathcal{D}$ is equivalent to setting the probability of sampling to zero for the target pair in negative sampling. Similar to noise in positive sampling, the threshold, t_-, for the proportion, q, of the data to be removed is obtained as

$$t_- = \mathrm{Percentile}(\{\rho^-_{u,i}\}_{(u,i)\in\mathcal{D}}, q).$$

Then, the probability that item j is sampled to $\mathcal{N}_{u,i}$ for positive pair (u,i) is set as

$$\Pr(j|u,i) = \begin{cases} 0 & (\rho^-_{u,i} \geq t_- \ \vee \ (u,j) \in \mathcal{S}) \\ \frac{1}{M_u} & \text{otherwise} \end{cases} \tag{4}$$

with $M_u = |\{i|Y_{u,i} = -1 \wedge \rho^-_{u,i} < t_-\}|$. These removals prevent extremely noisy data from being sampled in $\mathcal{B}$ and $\mathcal{N}_{u,i}$. We call these steps Confident Collaborative Metric Learning (CCML), and they are shown in Algorithm 1.

Algorithm 1 Confident Collaborative Metric Learning (CCML)

Require: Observed interaction data $\mathcal{S}$; set of users $\mathcal{U}$; set of items $\mathcal{I}$; latent dimension d; positive percentile p; negative percentile q; model f to predict $P(Y_{u,i} = 1)$; size of minibatch M_+; size of negative sample M_-; number of iterations T.

Ensure: Learned embeddings $\mathbf{v}_u, \mathbf{v}_i \in \mathbb{R}^d \ \forall u \in \mathcal{U}, \forall i \in \mathcal{I}$

$\mathcal{D} \leftarrow \mathcal{U} \times \mathcal{I} \setminus \mathcal{S}$
Train parameter Θ of model f using $\mathcal{S}$ and $\mathcal{D}$.
$\rho^+_{u,i} \leftarrow 1 - 2\sigma(f(u,i;\Theta)), \ \forall (u,i) \in \mathcal{S}$
$\rho^-_{u,i} \leftarrow 2\sigma(f(u,i;\Theta)) - 1, \ \forall (u,i) \in \mathcal{D}$
$t_+ \leftarrow \mathrm{Percentile}(\{\rho^+_{u,i}\}_{(u,i)\in\mathcal{S}}, p)$
$t_- \leftarrow \mathrm{Percentile}(\{\rho^-_{u,i}\}_{(u,i)\in\mathcal{D}}, q)$
$\mathcal{S}' \leftarrow \{(u,i)|(u,i) \in \mathcal{S} \wedge \rho^+_{u,i} \leq t_+\}$
Initialize $\mathbf{v}_u, \mathbf{v}_i, \forall u \in \mathcal{U}, \forall i \in \mathcal{I}$
for $k = 1, \cdots, T$ **do**
  $\mathcal{B} \leftarrow$ Sample a minibatch of size M_+ from $\mathcal{S}'$.
  for $(u,i) \in \mathcal{B}$ **do**
    $\mathcal{N}_{u,i} \leftarrow$ Sample a set of negative items of size M_- with probability defined in (4).
  end for
  Compute loss $\mathcal{L}^{\text{batch}}(\mathbf{v}_*, \mathcal{B}, \{\mathcal{N}_{u,i}\}_{(u,i)\in\mathcal{B}})$ defined in (2)
  Update $\mathbf{v}_*$ using $\frac{\partial \mathcal{L}^{\text{batch}}}{\partial \mathbf{v}_*}$ by employing Adam.
  Normalize $\mathbf{v}_*$ by $\mathbf{y}' = \frac{\mathbf{y}}{\max(\|\mathbf{y}\|,1)}$
end for
return $\mathbf{v}_*$

IV. EXPERIMENTS

In this section, we describe the experiments conducted to verify the performance of the proposed method on real-world datasets. In particular, we discuss whether the proposed method can improve the ranking metrics, as compared with the 2-stage negative sampling and weighted sampling.

These experiments were conducted from April to May 2021 using the Colaboratory's GPU and a regular memory instance, ensuring that the GPU was always an NVIDIA Tesla P100. The code for reproducing the results of the experiments is available here [1].

A. Datasets

The experiment uses the Yahoo! R3[2] and coat[3] datasets, which are used to evaluate recommendation systems with bias and noise removed. These datasets have been previously divided into training and test data. The training data contain the ratings of the items that are freely selected by each user, and the test data contain the ratings of the items that are randomly selected for each user. In other words, the training data contain noise and bias, similar to other datasets, whereas the test data contain almost no noise and bias. As these datasets have five ratings, we perform common preprocessing on these datasets. This enables us to simulate the positive and negative noise of implicit feedback on the training dataset. Then, we test whether we can learn accurately with noisy labels.

The preprocessing is carried out as follows:

1) First, we create the relevance data for implicit feedback by applying the following treatments to the five ratings of the training and test datasets.
   - $R_{u,i} = 1$ for (u,i) with a rating of 4 or 5.
   - $R_{u,i} = -1$ for (u,i) with a rating of 1 to 3.
   - $R_{u,i}$ is missing for (u,i) with no rating.

   We denote the relevance in the training and test data as $R^{\text{train}}_{u,i}$ and $R^{\text{test}}_{u,i}$, respectively.
2) Next, we create the noisy label, $Y_{u,i}$, observed in the training data using the following equation:

$$Y_{u,i} = \begin{cases} 1 & (R_{u,i}\text{is not missing}), \\ -1 & (R_{u,i}\text{is missing.}) \end{cases}$$

These preprocessing steps enable us to include two types of noise in the training data.

- Positive noise (false-positive): $Y_{u,i} = 1$ and $R^{\text{train}}_{u,i} = -1$.
- Negative noise (false-negative): $Y_{u,i} = -1$ and $R^{\text{train}}_{u,i} = 1$.

As we use $R^{\text{test}}_{u,i}$ for validation, we can verify the recommendation system using the data that contain no noise. Table I shows the dataset statistics after the preprocessing.

[1] https://colab.research.google.com/drive/1_4hegtR2BORSN8a9PjLTfTsHdiN55Z4p?usp=sharing
[2] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[3] https://www.cs.cornell.edu/~schnabts/mnar/

TABLE I
STATISTICS OF THE DATASETS AFTER PREPROCESSING.

Datasets	# Users	# Items	# Interactions
Yahoo! R3	15,400	1,000	311,704
coat	290	300	6960

B. Baselines and Proposed Model

In the experiment, we use the following previous methods as the baselines:

- uniform: Completely uniform sampling.
- popular: Weighted negative sampling by popularity.
- 2stage: 2-stage negative sampling.

The proposed methods are as follows:

- pos clean: Top p% noisy pairs are excluded from $\mathcal{S}$.
- neg clean: Top q% noisy pairs are excluded from $\mathcal{D}$.
- both clean (CCML): CML with both of the above.

In addition, we evaluate the following methods to understand the effect of elimination as an ablation study.

- pos weight: Positive sampling with $\Pr(u,i) \sim 1-\rho^{+}_{u,i}$.
- neg weight: Negative sampling with $\Pr(u,i) \sim 1-\rho^{-}_{u,i}$.
- both weight: CML with both of the above.

C. Metrics

We use three ranking metrics that represent the accuracy of predicting the ranking of $R^{\text{test}}_{u,i}=1$ items for each user. We compute the ranking by utilizing the distance between the user vector and item vector calculated by the CML model. On the basis of the previous studies [14], [19], that have used the same datasets, we adopt the following three ranking indices:

$$\text{DCG@K} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}^{\text{test}}_u|}\sum_{i\in\mathcal{S}^{\text{test}}_u}\frac{\mathbb{I}\left\{\widehat{Z}_{u,i}\le K\right\}}{\log\left(\widehat{Z}_{u,i}+1\right)}$$

$$\text{MAP@K} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}^{\text{test}}_u|}\sum_{i\in\mathcal{S}^{\text{test}}_u}\sum_{k=1}^{K}\frac{\mathbb{I}\left\{\widehat{Z}_{u,i}\le k\right\}}{k}$$

$$\text{Recall@K} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{S}^{\text{test}}_u|}\sum_{i\in\mathcal{S}^{\text{test}}_u}\mathbb{I}\left\{\widehat{Z}_{u,i}\le K\right\}$$

Here, $\widehat{Z}_{u,i}$ represents the predicted ranking of item i for user u, and $\mathcal{S}^{\text{test}}_u=\{i|R^{\text{test}}_{u,i}=1\}$. In addition, we use nDCG@K to denote the normalized value of DCG@K at the ideal prediction rank. Furthermore, we remove the users for whom $|\mathcal{S}^{\text{test}}_u|=0$ because we have judged that the recommendation system cannot improve the experience of these users. The previous studies that have used the same datasets set the scores for such users as 0. Thus, our experimental scores are higher overall, but we do not compromise the order.

D. Hyperparameters

We set the learning rate as 1e-3, the dimension, d, of $\mathbf{v}_*$ as 10, and $M_+=256$ for the training data. For the Yahoo dataset, we use 50k minibatch iterations and $M_-=10$. For the coat dataset, we use 10k minibatch iterations and $M_-=5$. We use LMF to estimate the noise rate. We implement LMF in a unified manner by changing the CML loss from equation (1) to equation (3). In LMF, we perform 20k and 3k minibatch iterations for the Yahoo and coat datasets, respectively. Finally, we set the percentage of data to be eliminated as $p=q=5\%$ in the proposed method. As it is difficult to prepare two or more test datasets, hyperparameter tuning will be considered in future work.

E. Results

Table II shows the scores of the three ranking metrics for the two values of k and runtime (including noise rate estimation) for the baseline and proposed methods on the Yahoo! R3 dataset. Moreover, Table III shows the scores and runtime for the coat dataset. The bold and underlined values indicate the highest and second-highest scores, respectively. These scores are the averages of five training runs performed on the training dataset, where we randomly selected 90% of the positive interactions and calculated each metric. The results show that weighting or cleaning based on the estimated noise rate considerably improves the scores of all the metrics over the uniform and existing techniques (popular, and 2-stage).

Figure 2 and Figure 3 depict the mean and confidence intervals of the nDCG@3 and Recall@5 scores for each method over the learning process, respectively. As shown in these figures, the confidence interval at the end of training is small and the performance improvement is significant. Furthermore, these results show that all the methods converge at the end of the iterations, and no further improvement is expected even if the training time is increased. Therefore, based on the runtime results in Table II and Table III, we can conclude that the performance of the proposed method can be improved by using approximately twice the computational time of *uniform*.

More specifically, it is evident that *neg clean* and *both clean* are the best and second-best methods on the Yahoo! R3 dataset, whereas *neg weight* and *both clean* are the best and second-best methods on the coat dataset, respectively. By contrast, *neg weight*, which is the best on the coat dataset, is not better than any of the cleaning methods (*pos clean*, *neg clean*, and *both clean*) on the Yahoo! R3 dataset; moreover, *neg clean*, which is the best on the Yahoo! R3 dataset, is not the best method on the coat dataset. Thus, the most stable and efficient method is the *both clean* method. In other words, in many situations, it is more effective to use both positive and negative noise to remove extremely noisy data instead of weighting them.

As compared to uniform, the worse scores for popular and 2-stage on the Yahoo! R3 dataset are caused by false-positive noise. In previous experiments that highlighted the effectiveness of methods using item popularity, there was no

TABLE II
RANKING METRICS AND RUN TIME FOR EACH METHOD ON YAHOO! R3 DATASET.

	nDCG@3	MAP@3	Recall@3	nDCG@5	MAP@5	Recall@5	runtime [s]
uniform	0.494	0.550	0.523	0.567	0.564	0.702	3.90×10^2
popular	0.450	0.493	0.502	0.534	0.516	0.704	4.26×10^2
2stage	0.416	0.459	0.464	0.505	0.487	0.680	5.05×10^2
pos clean	0.551	0.604	0.582	0.620	0.611	0.755	7.21×10^2
neg clean	**0.559**	**0.612**	**0.590**	**0.628**	**0.620**	**0.763**	7.43×10^2
both clean (CCML)	**0.559**	**0.612**	**0.590**	0.626	0.619	0.760	7.53×10^2
pos weight	0.539	0.591	0.572	0.609	0.600	0.746	6.02×10^2
neg weight	0.530	0.585	0.559	0.599	0.594	0.730	6.51×10^2
both weight	0.541	0.596	0.571	0.613	0.605	0.748	6.60×10^2

TABLE III
RANKING METRICS AND RUN TIME FOR EACH METHOD ON COAT DATASET.

	nDCG@3	MAP@3	Recall@3	nDCG@5	MAP@5	Recall@5	runtime [s]
uniform	0.271	0.355	0.219	0.312	0.371	0.337	0.61×10^2
popular	0.253	0.326	0.209	0.296	0.349	0.333	0.63×10^2
2stage	0.296	0.390	0.234	0.347	0.416	0.375	0.62×10^2
pos clean	0.340	0.448	0.281	0.393	0.462	0.432	1.19×10^2
neg clean	0.346	0.449	0.290	0.397	0.467	0.438	1.06×10^2
both clean (CCML)	0.346	0.450	0.290	**0.402**	**0.471**	**0.450**	1.18×10^2
pos weight	0.341	0.446	0.282	0.393	0.466	0.432	1.00×10^2
neg weight	**0.347**	**0.453**	**0.294**	0.394	0.469	0.439	1.02×10^2
both weight	0.347	0.445	0.289	0.394	0.467	0.431	1.02×10^2

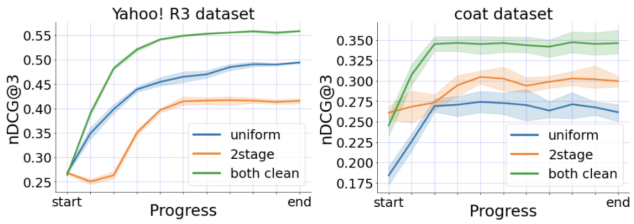

Fig. 2. Means and confidence intervals of nDCG@3 for the learning progress of each method. The horizontal axis represents the percentage of progress from the beginning to the end of CML training. (left): results on the Yahoo! R3 dataset. (right): results on the coat dataset.

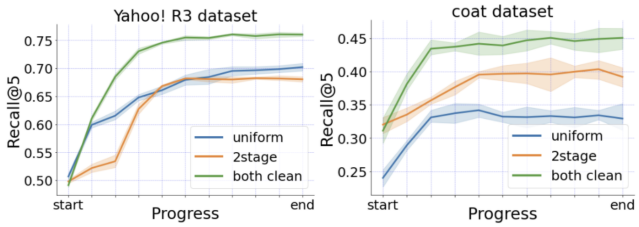

Fig. 3. Means and confidence intervals of Recall@5 for the learning progress of each method. The horizontal axis represents the percentage of progress from the beginning to the end of CML training. (left): results on the Yahoo! R3 dataset. (right): results on the coat dataset.

noise in positive feedback. Therefore, it was assumed that the correct popularity was available. However, only noisy feedback is available in this experiment; therefore, the correct popularity was not calculated. Thus, popularity considers noisy items as overly popular. As a result, the model heavily samples noisy user–item pairs during negative sampling. This result suggests that it is not adequate to rely only on implicit feedback to calculate popularity in the real world.

F. Explainability of embeddings

This subsection presents the determination of whether the performance improvement of the proposed method, discussed in the previous subsection, is due to the capturing of user–user and item–item relationships. For this purpose, we evaluate whether users and items with similar attributes are embedded at short distances. This experiment uses the user's gender and the item's target gender information in the coat dataset as attributes. It should be noted that the item in this dataset represents clothing. Figure 4 shows scatter plots of the user embeddings learned by *uniform* and *both clean*, compressed into two dimensions by t-SNE [33] and color-coded by gender. Figure 5 also presents scatter plots of the item embedding color-coded by target gender. There are few interactions across genders in clothing, and thus, it is easy to express attributes; nevertheless, these figures show that the embedding generated by *uniform* is not ideal in certain areas.

To quantify this evaluation, we used the SVM [34] to evaluate how well it can classify attributes. To this end, we computed the error rate of classifying the labels of the attributes using the user and item embeddings as features. As SVM is a classifier that predicts labels based on the distance between the samples in a vector space, it is a suitable algorithm for evaluating the embeddings produced by CML.

Table IV shows the results of evaluating the error rate of classification via 5-fold cross-validation. The two columns on the left are for the user classification, and the two columns on the right are for the item classification. On the left of each is the average error rate and on the right is the maximum error rate (worst case). These results show that the error rate of

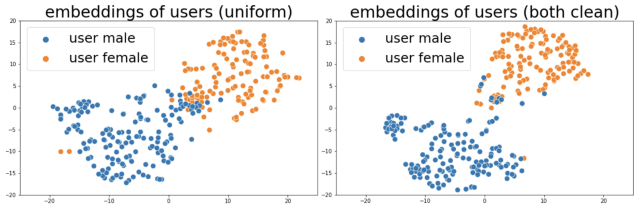

Fig. 4. Scatterplot of user embeddings color-coded by gender. (Left): User embedding when using *uniform*. (Right): User embedding when using *both clean*.

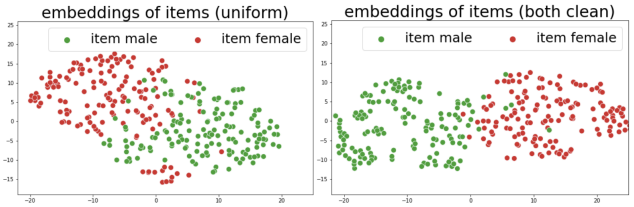

Fig. 5. Scatterplot of item embeddings color-coded by target gender. (Left): Item embeddings when using *uniform*. (Right): Item embeddings when using *both clean*.

TABLE IV
ERROR RATE (%) WHEN SVM CLASSIFIES LABELS OF ATTRIBUTES USING EMBEDDINGS AS FEATURES.

	User		Item	
	Avg.	Max.	Avg.	Max.
uniform	7.24	12.07	1.67	5.00
both clean (CCML)	**5.52**	**8.62**	**0.67**	**1.67**

both clean is significantly improved, as compared to that of *uniform*, by approximately 3/4 for users and 1/2 for items. Notably, in the worst case, the improvement is even more significant. Thus, the improvement in the ranking metrics, discussed in the previous subsection, can be explained by better capturing the user–user and item–item relationships. This result implies that the proposed method, CCML, can meet the requirements of modern recommendation systems.

V. CONCLUSION

This study proposes a method for CML that considers comprehensive noise in implicit feedback. The method, CCML, first estimates the noise rate and then performs CML by excluding the top user–item pairs with a high noise rate. To account for noise in positive and negative labels caused by users and items, we adopt a model used in general recommendation systems to estimate the noise rate. In addition, we address noise by deleting data instead of weighted sampling because the inclusion of noise has a significant negative effect on learning. The effectiveness of this method is supported by a method for image classification in noisy data. The proposed method significantly outperforms existing methods in experiments conducted using train set with noise and test set with negligible or no noise.

The data deletion ratios, p and q, are considered as hyperparameters in this study. It is difficult to tune them because extremely high ratios result in information loss and extremely low ratios lead to noise. In future, we will examine a method for determining the optimal ratios on the basis of this trade-off.

REFERENCES

[1] J.-T. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang, "Embedding-based retrieval in facebook search," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2553–2561.

[2] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; Data Mining*, 2018, pp. 311–320.

[3] D. Guo, J. Xu, J. Zhang, M. Xu, Y. Cui, and X. He, "User relationship strength modeling for friend recommendation on instagram," *Neurocomputing*, vol. 239, pp. 9–18, 2017.

[4] Y. M. Brovman, M. Jacob, N. Srinivasan, S. Neola, D. Galron, R. Snyder, and P. Wang, "Optimizing similar item recommendations in a semi-structured marketplace to maximize conversion," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 199–202.

[5] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 193–201.

[6] L. Yu, C. Zhang, S. Pei, G. Sun, and X. Zhang, "Walkranker: A unified pairwise ranking model with multiple relations for item recommendation," in *AAAI*, 2018, pp. 2596–2603.

[7] C.-M. Chen, C.-J. Wang, M.-F. Tsai, and Y.-H. Yang, "Collaborative similarity embedding for recommender systems," in *The World Wide Web Conference*, 2019, pp. 2637–2643.

[8] M. Campo, J. Espinoza, J. Rieger, and A. Taliyan, "Collaborative metric learning recommendation system: Application to theatrical movie releases," *arXiv preprint arXiv:1803.00202*, 2018.

[9] Y. Tay, L. A. Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 729–739.

[10] X. Zhou, D. Liu, J. Lian, and X. Xie, "Collaborative metric learning with memory network for multi-relational recommender systems," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 4454–4460.

[11] C. Park, D. Kim, X. Xie, and H. Yu, "Collaborative translational metric learning," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 367–376.

[12] L. Vinh Tran, Y. Tay, S. Zhang, G. Cong, and X. Li, "Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 609–617.

[13] V.-A. Tran, G. Salha-Galvan, R. Hennequin, and M. Moussallam, "Hierarchical latent relation modeling for collaborative metric learning," *arXiv preprint arXiv:2108.04655*, 2021.

[14] Y. Saito, "Unbiased pairwise learning from biased implicit feedback," in *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, 2020, pp. 5–12.

[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173–182.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI '09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[17] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 191–198.

[18] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 335–344.

[19] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, "Unbiased recommender learning from missing-not-at-random implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 501–509.
[20] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 263–272.
[21] H. Lu, M. Zhang, and S. Ma, "Between clicks and satisfaction: Study on multi-phase user preferences and satisfaction for online news reading," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 435–444.
[22] W. Wang, F. Feng, X. He, L. Nie, and T.-S. Chua, "Denoising implicit feedback for recommendation," in *WSDM 2021: The 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 373–381.
[23] W. Wang, F. Feng, X. He, H. Zhang, and T.-S. Chua, "Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue," in *SIGIR 2021: 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.
[24] W. Yu and Z. Qin, "Sampler design for implicit feedback data by noisy-label robust learning," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 861–870.
[25] N. Wang, Z. Qin, X. Wang, and H. Wang, "Non-clicks mean irrelevant? propensity ratio scoring as a correction," in *WSDM 2021: The 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 481–489.
[26] V.-A. Tran, R. Hennequin, J. Royo-Letelier, and M. Moussallam, "Improving collaborative metric learning with efficient negative sampling," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 1201–1204.
[27] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, 2021.
[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, vol. 26, 2013, pp. 3111–3119.
[29] O. Barkan and N. Koenigstein, "Item2vec: Neural item embedding for collaborative filtering." in *RecSys Posters*, 2016.
[30] C. C. Johnson, "Logistic matrix factorization for implicit feedback data," *Advances in Neural Information Processing Systems*, vol. 27, no. 78, pp. 1–9, 2014.
[31] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 995–1000.
[32] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR : International Conference on Learning Representations*, 2015.
[33] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.
[34] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.